PATENT ABSTRACTS OF JAPAN

(11)Publication number : 09-022361
(43)Date of publication of application : 21.01.1997

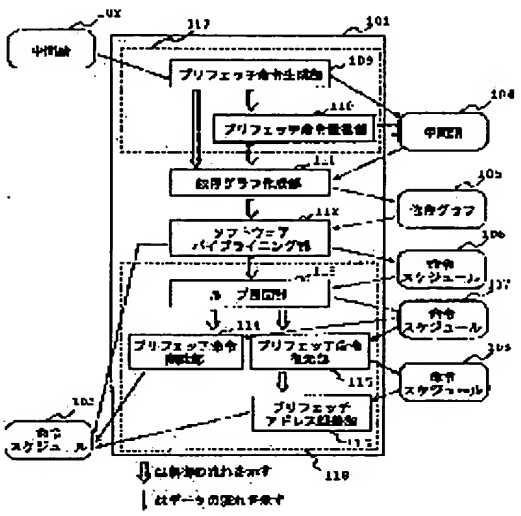
(51)Int.Cl. G06F 9/45

(21)Application number : 07-170674
(22)Date of filing : 06.07.1995

(71)Applicant : HITACHI LTD
(72)Inventor : NISHIYAMA HIROYASU
KIKUCHI SUMIO

(54) DATA PREFETCH METHOD

(57)Abstract:
PROBLEM TO BE SOLVED: To reduce a wait caused by memory reference and a wait caused by the depending relation between instructions by generating a dependency graph having a branch between a generated prefetch instruction and a correspondent memory reference instruction.
SOLUTION: A prefetch instruction generating part 109 consists of an intermediate word 104, to which the prefetch instruction is added, by inputting an intermediate word 102 and generating the prefetch instruction concerning the memory reference instruction having the high possibility of cache error occurrence among the memory reference instructions contained in the intermediate word at the main body of a loop. Next, at a dependency graph preparing part 111, a dependency graph 105 is prepared by inputting the intermediate word 104 containing the prefetch instruction. Afterwards, software pipelining is applied to the dependency graph 105 by a software pipelining part 112 and a software pipelined instruction schedule 103 is provided. Thus, the prefetch instruction can be scheduled so that the wait caused by memory reference can be hidden.



LEGAL STATUS

[Date of request for examination]	19.10.1998
[Date of sending the examiner's decision of rejection]	06.02.2001
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]	
[Date of final disposal for application]	
[Patent number]	3218932
[Date of registration]	10.08.2001
[Number of appeal against examiner's decision of rejection]	2001-02982

[Date of requesting appeal against examiner's decision of rejection] 01.03.2001

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号

特開平9－22361

(43)公開日　平成９年(1997)１月21日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ　9/45		9189－5B	Ｇ０６Ｆ　9/44	３２２Ｇ	

審査請求　未請求　請求項の数６　ＯＬ　（全 16 頁）

(21)出願番号　特願平7－170674

(22)出願日　平成７年(1995)７月６日

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地

(72)発明者　西山　博泰
神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内

(72)発明者　菊池　純男
神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内

(74)代理人　弁理士　小川　勝男

(54)【発明の名称】　データプリフェッチ方法

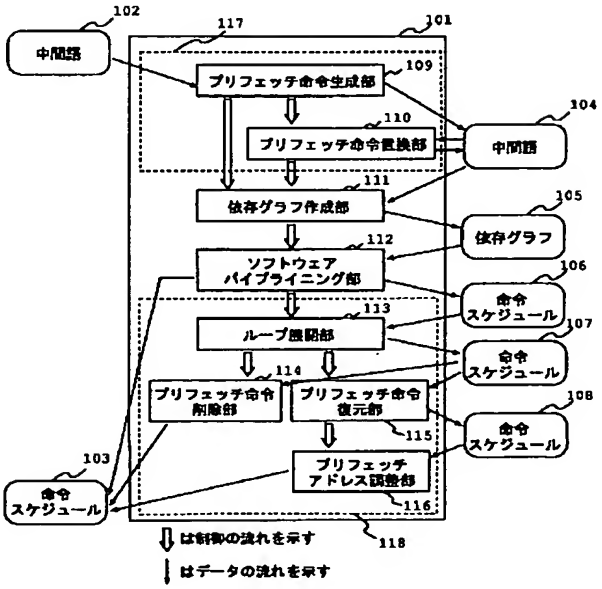
(57)【要約】

【目的】他の命令の実行と同時に主記憶からキャッシュメモリにデータを移動する機能を持つプリフェッチ命令を効率良くスケジュールする。

【構成】　プリフェッチ命令とメモリ参照命令の間に依存を持つ依存グラフを作成してソフトウェアパイプライニングを適用するか、あるいは、さらに、構成されたスケジュールのカーネル部を展開して冗長なプリフェッチ命令を削除するか、あるいは、複数のプリフェッチ命令をまとめて仮想的なプリフェッチ命令に置き換えて依存グラフを作成したものにソフトウェアパイプライニングを適用して構成されたスケジュールのカーネル部を展開した仮想的なプリフェッチ命令を元のプリフェッチ命令に置き換えてプリフェッチ命令をスケジュールする。

【効果】　キャッシュメモリ参照ミスと命令間の遅延による待ちを減少させ、計算機プログラムの実行の高速化に有効である。

図１



【特許請求の範囲】
【請求項１】他の命令の実行と並行して主記憶からキャッシュメモリにデータを転送するプリフェッチ命令を有する計算機上で実行されるプログラムをコンパイルするコンパイラにおいて、
(a)プログラム中のループのソースプログラムを中間コードへ変換し、
(b)前記変換された中間コードに基づいて、プリフェッチ命令とメモリ参照を行なう命令との間に枝を持つ依存グラフを作成し、
(c)ループの異なる繰り返しを重ね合わせて実行することにより命令間の遅延が可能な限り隠蔽されるように命令をスケジューリングするソフトウェアパイプライニングを、プリフェッチ命令とメモリ参照を行なう命令の間の遅延をプリフェッチに要するサイクル以上にするという制約を設けて、前記依存グラフに適用することにより、命令スケジューリングを行なうことを特徴とするデータプリフェッチ方法。
【請求項２】前記方法の前記ステップ(c)において、
(c１)スケジュールされたループ本体を複数回展開し、
(c２)前記展開されたループ本体中の冗長なプリフェッチ命令を削除することを特徴とする請求項１記載のデータプリフェッチ方法。
【請求項３】他の命令の実行と並行して主記憶からキャッシュメモリにデータを転送するプリフェッチ命令を有する計算機上で実行されるプログラムをコンパイルするコンパイラにおいて、
(a)プログラム中のループのソースプログラムを中間コードへ変換し、
(b)プログラム中のループに含まれる複数のプリフェッチ命令を１つの仮想的なプリフェッチ命令に置き換え、
(c)前記中間コードに基づいて、前記仮想的なプリフェッチ命令とメモリ参照を行なう命令との間に枝を持つ依存グラフを作成し、
(d)ループの異なる繰り返しを重ね合わせて実行することにより命令間の遅延が可能な限り隠蔽されるように命令をスケジューリングするソフトウェアパイプライニングを、前記依存グラフに適用して命令スケジューリングを行ない、
(e)前記命令スケジュールを展開することを特徴とするデータプリフェッチ方法。
【請求項４】前記方法の前記ステップ(e)において、
(e１)得られたスケジュールを複数回展開し、
(e２)前記展開された仮想的なプリフェッチ命令を元の複数のプリフェッチ命令に置き換えることを特徴とする請求項３記載のデータプリフェッチ方法。
【請求項５】前記方法の前記ステップ(e)は、さらに、前記置き換えられたプリフェッチ命令が参照するアドレスを、前記プリフェッチ命令によるデータの転送が完了するのに十分先の繰り返しで参照されるアドレスを参照するように調節することを特徴とする請求項４記載のデータプリフェッチ方法。
【請求項６】他の命令の実行と並行して主記憶からキャッシュメモリにデータを転送するプリフェッチ命令を有する計算機上で実行されるプログラムをコンパイルするコンパイラにおいて、
前記プリフェッチ命令を含んだループをソフトウェアパイプライニングによってスケジュールすることを特徴とするデータプリフェッチ方法。
【発明の詳細な説明】
【０００１】
【産業上の利用分野】本発明はデータプリフェッチ方法に関し、さらに詳しくは、ループに対してプリフェッチ命令をスケジュールすることによりデータのプリフェッチを行ないプログラムの実行時間を短縮するために有効なコンパイル方法に関する。
【０００２】
【従来の技術】計算機上でのプログラムの実行においては、命令間の依存関係によって生じる待ちと，メモリ参照によって生じる待ちが、実行時間の多くを占めている。
【０００３】ループ内の命令間の依存関係による待ちは、ソフトウェアパイプライニングと呼ばれるスケジューリング手法によってかなり低減できる。ソフトウェアパイプライニングとは，例えば、ラマクリシュナン、ソフトウエア パイプライニング イン ＰＡ－ＲＩＳＣ コンパイラ、ヒューレットパッカードジャーナル、３９－４５頁、１９９２年(S. Ramakrishnan, Software pipelining in PA-RISC compilers, Hewlett-Packard Journal, pp.39-45,1992)に述べられているように、ループの異なる繰り返しを重ね合わせて実行することにより命令間の依存関係による待ちを減少させ、命令実行の並列度を高める方式である。ソフトウェアパイプライニングが適用されたループでは、ループの実行開始前にプロローグと呼ばれる初期化処理を行なうコードを実行し、カーネルと呼ばれるコードを繰り返すことによりループ本体の実行を行ない、ループの実行を終了した時点でエピローグと呼ばれるコードを実行して終了処理を行い、かつ、1回前の繰り返しの完了を待たずに後続する繰り返しの実行を開始することをその特徴とする。
【０００４】命令間の依存関係による待ちに対して、メモリ参照による待ちはソフトウェアによる手法のみでは低減することが難しい。そのため、多くの計算機システムにおいては、メモリ参照による待ちを減少するために、キャッシュと呼ばれる高速少容量の記憶装置を主記憶とプロセッサの間に配置し、最近参照されたデータをキャッシュ上に置いて高速に参照できるようにしている。ただし、キャッシュを付加した場合においても、データの再利用がない場合などには、キャッシュミスが発生した場合の待ちは避けられない。

【０００５】そこで、例えば、モウリー他、デザイン アンド エバリュエーション オブ アコンパイラ アルゴリズム フォ プリフェッチング、プロシーディング オブ ザ５ インターナショナル コンファレンス オン アーキテクチュアル サポートフォ プログラミング ランゲージ アンド オペレーティング システム、６２－７３頁、１９９２年(T.C.Mowry, et al., Design and Evaluation of a Compiler Algorithm for Prefetching, Proceedings of the 5th International Conference on Architectural Support for Programming Languages and Operating Systems, pp.62-73,1992)に述べられているように、主記憶からキャッシュにデータを先行的に移動(プリフェッチ)する命令を利用することによって、メモリ参照による待ちを低減することが試みられている。

【０００６】

【発明が解決しようとする課題】上記従来技術では、プリフェッチ命令をスケジュールするための方式として，プリフェッチ命令の遅延をループ本体の最短パスの長さで割った値以上の最小の整数回分だけ前の繰り返し前にプリフェッチ命令が発行されるようにソフトウェアパイプライニングを適用とすると述べられているが，実現の詳細については明らかにされていない。

【０００７】そこで、本発明の目的は、プリフェッチ命令を含んだループにおけるプログラム実行時のメモリ参照による待ちと命令間の依存関係による待ちを低減できる効率的な命令スケジューリング方法を提供することにある。

【０００８】

【課題を解決するための手段】本発明の目的を達成するため、本発明では、プログラムのコンパイルの際に、以下に示す３種の方式のいずれかにより、プログラム中のループに対してプリフェッチ命令のスケジューリングを行なう。

【０００９】キャッシュへのデータのプリフェッチはデータの値に変更を加えない。このため、一般的なデータの定義と使用の関係に従うと、キャッシュへのデータのプリフェッチと、ロード命令やストア命令によるメモリの参照との間には依存関係は存在しない。ただし、メモリ参照による待ちを隠蔽するには、プリフェッチ命令によるキャッシュへのデータの転送の完了後にメモリ参照命令を発行しなければならないという制約から、プリフェッチ命令とメモリ参照を行なう命令の間に、暗黙の依存関係があるとみなすと既存のスケジューリング方式をそのまま適用でき都合が良い。そこで、方式１では以下のように、プリフェッチ命令とメモリ参照命令の間に依存関係を設けてスケジューリングを行なう。

【００１０】方式１:

(1)キャッシュミスを生じると予想されるメモリ参照命令のそれぞれに対してプリフェッチ命令を生成する。

【００１１】(2)上記(1)で生成したプリフェッチ命令と、対応するメモリ参照命令の間に枝を持つ依存グラフを生成する。この際、プリフェッチ命令が発行されてから、プリフェッチ命令によるキャッシュへのデータ転送に要するサイクル数を経た後でメモリ参照命令が発行されるよう、プリフェッチ命令とメモリ参照命令との間の遅延を、プリフェッチ命令によるキャッシュへのデータ転送に要するサイクル数以上に設定する。

【００１２】(3)上記(2)で構成した依存グラフに対して、ソフトウェアパイプライニングを適用し、命令スケジュールを得る。上記のように、ソフトウェアパイプライニングとはループの異なる繰り返しを重ね合わせて実行することにより、命令間の依存関係による待ちを減少させる方式であり、これを上記(2)で構成した依存グラフへ適用することによって、プリフェッチ命令と対応するメモリ参照命令の間を十分離すことができる。

【００１３】主記憶からキャッシュへのデータの転送単位が一般的には32バイトや128バイトといった単位であるのに対して、ループ中での配列などの参照は4バイトや8バイトといった、より小さな単位で行なわれることが多い。このため、ループ中で配列などに対して連続的なメモリ参照を行なう場合には、一度のプリフェッチ命令によって、複数回分の繰り返しの実行で参照するデータを主記憶からキャッシュへ移動することができる場合が多い。すなわち、一度のプリフェッチ命令によってＮ回分の繰り返しの実行で参照するデータを主記憶からキャッシュへ移動することができるとすると、プリフェッチ命令の発行はＮ回に１度行なえばよいことになる。

【００１４】方式１で生成したスケジュールでは、繰り返し１回毎にプリフェッチ命令を発行するため、冗長なプリフェッチ命令が多数発行されることになってしまう。そこで、方式２および方式３では、ループ本体を展開し冗長なプリフェッチ命令がなるべく発行されないようにプリフェッチ命令をスケジュールする。

【００１５】まず、方式２では、上記方式１の(1)から(3)までの処理によって生成されたソフトウェアパイプライン化されたプリフェッチ命令を含むループのカーネル部を展開し、冗長なプリフェッチ命令を取り除くことで無駄なプリフェッチ命令が発行されないようにする。

【００１６】方式２:

(4)プリフェッチ命令の発行は、一回のプリフェッチ命令でプリフェッチできるデータ数をＮとすると、Ｎ回の繰り返し毎に行なえばよいので、まず、上記(3)で構成されたソフトウェアパイプライン化されたスケジュールのカーネル部を展開し、その展開数がNの倍数となるようにする。

【００１７】(5)上記(4)の展開コードでは、カーネル部がＮの倍数回展開され、展開されたカーネル部の１回の繰り返しで元のループのＮの倍数回分の繰り返しが実行されることになる。そこで、展開されたコードから、Ｎ回に一回プリフェッチ命令が発行されるよう、冗長な

プリフェッチ命令を削除すれば、無駄なプリフェッチ命令の発行がなくなる。

【００１８】方式２では、プリフェッチ命令にソフトウェアパイプライニングを適用した後で無駄なプリフェッチ命令を削除するので、プリフェッチ命令を削除したことによって命令間の距離がソフトウェアパイプライニングを適用した際に期待したものよりも短くなり、これによって命令間の依存などによる待ちが生じやすくなってしまう可能性がある。

【００１９】そこで、方式３では、ループをソフトウェアパイプライン化してスケジュールを得た後でカーネル部を展開するすることを考慮して、まず、複数のプリフェッチ命令をまとめて１つの仮想的なプリフェッチ命令に置き換え、この仮想的なプリフェッチ命令を含んだ依存グラフを作成する。ただし、方式１や方式２の場合と異なり、方式３では仮想的なプリフェッチ命令と、対応するメモリ参照命令との間に依存は設けなくともよい。

【００２０】次に、ソフトウェアパイプライニングを依存グラフに適用し、ソフトウェアパイプライン化されたスケジュールを得、カーネル部の展開数が一度のプリフェッチ命令でプリフェッチできるデータ数の倍数となるように必要に応じてループを展開する。展開された仮想的なプリフェッチ命令を元のプリフェッチ命令に置き換え、プリフェッチ命令が，対応するメモリ参照命令よりも十分前の繰り返しで発行されるよう、プリフェッチ命令が参照するアドレスを調節する。

【００２１】これにより、方式２で命令を削除したことによって発生した命令間の依存を減少させる。

【００２２】方式３：

(1)キャッシュミスを生じると予想されるメモリ参照命令それぞれに対してプリフェッチ命令を生成する。

【００２３】(2)上記(1)で生成したプリフェッチ命令を複数個組にして仮想的なプリフェッチ命令に置き換える。

【００２４】(3)元のループボディーの命令と上記(2)で生成した仮想的なプリフェッチ命令からなる依存グラフを作成して、ソフトウェアパイプライニングを適用する。依存グラフの作成を行なう場合には、仮想的なプリフェッチ命令と、対応するメモリ参照命令との間の依存は考えなくとも良い。

【００２５】(4)上記(3)で構成されたカーネル部の展開数が、一度のプリフェッチ命令でデータをプリフェッチできる繰り返し数の倍数となるように必要に応じてループの展開を行なう。展開後のスケジュールにおいて、仮想的なプリフェッチ命令は元のプリフェッチ命令を挿入する候補となる命令スロットを表している。

【００２６】(5)上記(4)の展開コード中にスケジュールされた仮想的なプリフェッチ命令を、元のプリフェッチ命令に置き換える。この置き換えは、一度のプリフェッチ命令でデータをプリフェッチできる繰り返し数の倍数毎に、同一のプリフェッチ命令が発行されるようにする。これによって無駄なプリフェッチ命令の発行が抑制される。

【００２７】(6)上記(5)で置き換えたプリフェッチ命令が参照するアドレスをプリフェッチ命令によるデータの転送が完了するのに十分先の繰り返しで参照されるデータのアドレスとする。

【００２８】

【作用】本発明の方法によれば、メモリの参照が連続的でない場合には、方式１によってプリフェッチ命令と対応するメモリ参照命令の間を十分離してソフトウェアパイプライニングを適用することができる。また、メモリの参照が連続的である場合には、方式２によってソフトウェアパイプライニングを適用した後で命令の削除を行なうか、方式３によって複数のプリフェッチ命令を仮想的なプリフェッチ命令に置き換えてソフトウェアパイプライニングを適用し、その後、元のプリフェッチ命令に復元することで無駄なプリフェッチ命令を発行を抑制し効率的にスケジュールすることができる。これにより本発明の目的を達成できる。

【００２９】

【実施例】以下、図面を参照しながら本発明の一実施例について説明する。

【００３０】図２は本発明の方法を実施する計算機の１つの例である。この例では、CPU201上で動作するコンパイラが外部記憶装置202からソースコード203を読み込み、これをオブジェクトコード204へ変換し外部記憶装置202へ格納する。

【００３１】図３は、本発明によるデータのプリフェッチ方法を適用する計算機の１つの例である。CPU301で通常のメモリ参照命令の実行を行なう場合には、まず、キャッシュ 302に参照対象のデータがあるかどうかを調べ、キャッシュ302にデータが存在すればそのデータを参照し、キャッシュ302に参照対象のデータが存在しなければ主記憶303上の当該データを参照すると共に、当該データの属するキャッシュブロックをキャッシュ302に置く。キャッシュの参照は主記憶の参照に比べて高速であり、参照対象のデータがキャッシュにあればメモリ参照によって発生する待ちを低減できる。

【００３２】プリフェッチ命令は、他の命令の実行と同時に主記憶303からキャッシュ302へ参照対象のデータが属するキャッシュブロックを移動する命令である。主記憶303からキャッシュ302へキャッシュブロックを移動するのに十分なサイクル数だけ前にプリフェッチ命令を発行しておけば、主記憶303からキャッシュ302へのデータの転送を行なっている間に他の命令を実行できるので、当該データを参照するための待ちは無くなる。

【００３３】図１に本発明の一実施例の構成を示す。図１において、スケジューリング処理部101はループ本体に対する中間語102を入力し、プリフェッチ命令を含ん

だメモリ間の依存およびメモリ参照による待ちの少ない命令スケジュール103を出力する。処理117および118は本発明に特徴的な処理であり、処理117ではプリフェッチ命令の生成と、スケジューリングの前処理を行ない、処理118では無駄なプリフェッチ命令の除去やプリフェッチアドレスの調整などの後処理を行なう。

【００３４】始めに、方式１によりループをスケジュールする場合の実施例を示す。図４は方式１によってプリフェッチ命令のスケジュールを行なう場合の命令スケジューラの構成図である。方式１では、プリフェッチ命令生成部109が中間語102を入力し、ループ本体の中間語に含まれるメモリ参照命令のうち、キャッシュミスを起こす可能性の高いものについてプリフェッチ命令を生成して、プリフェッチ命令を加えた中間語104を構成する。

【００３５】ここで、あるメモリ参照命令に対してキャッシュミスが起きる可能性があるかどうかについては、例えば、モウリー他、デザイン アンド エバリュエーションオブ ア コンパイラ アルゴリズム フォ プリフェッチング、プロシーディングオブ ザ ５ インターナショナル コンファレンス オン アーキテクチュアル サポート フォ プログラミング ランゲージ アンド オペレーティング システム、６２－７３頁、１９９２年(T.C. Mowry他, Design and Evaluation of a Compiler Algorithm for Prefetching, Proceedings of the 5th International Conference on Architectural Support for Programming Languages and Operating Systems, pp.62-73, 1992)で述べられている公知技術やプログラムの実行のトレースを用いて推測することができる。生成するプリフェッチ命令がプリフェッチするアドレスは、対応するメモリ参照命令と同じとする。

【００３６】すなわち、ループ中のロード命令、

LOAD　X[i]

がキャッシュミスを起こしやすいとすると、同一要素をプリフェッチする命令、

FETCH　X[i]

を作り、これを中間語に加える。

【００３７】次に、依存グラフ作成部111では、プリフェッチ命令を含んだ中間語104を入力として依存グラフ105を作成する。この際、プリフェッチ命令と対応するメモリ参照命令との間に必要な遅延が主記憶からキャッシュへキャッシュブロックを転送するのに要する時間以上であることを表す枝を、プリフェッチ命令と対応するメモリ参照命令との間に設ける。次に、依存グラフ105に対してソフトウェアパイプライニング部112 でソフトウェアパイプライニングを適用して、ソフトウェアパイプライン化された命令スケジュール 103を得る。

【００３８】以上のように、プリフェッチ命令と対応するメモリ参照命令の間に、必要な遅延が主記憶からキャッシュへキャッシュブロックを転送するのに要する時間以上であることを表す枝を設けた依存グラフを作成することにより、ソフトウェアパイプライニングを適用する際に、プリフェッチ命令とそれに対応するメモリ参照命令との間が、主記憶からキャッシュへキャッシュブロックを転送するのに要する時間だけ離れることが保証されるので、メモリ参照による待ちが隠蔽されるようプリフェッチ命令をスケジュールすることができる。

【００３９】上記説明におけるプリフェッチ命令生成部109を、図7に示す動作フローチャートを参照しつつ説明する。まず、ステップ701では処理すべきメモリ参照命令が残っているかどうかを判定し、あればステップ702へ制御を移し、なければ処理を終了する。 ステップ702では、処理すべきメモリ参照命令を選択して変数MIに記憶する。ステップ703では、MIに記憶されたメモリ参照命令がキャッシュミスを起こす可能性が高いかどうかを判定し、キャッシュミスを起こす可能性が高いならばステップ704 へ制御を移し、低い場合にはステップ701へ制御を移して次のメモリ参照命令を処理する。ステップ704ではMIに記憶したメモリ参照命令と同じアドレスを参照するプリフェッチ命令を作成する。

【００４０】次に、方式２によりループをスケジュールする場合の実施例を説明する。方式２では、方式１の処理に加えて、以下のような処理を行なう。まず、方式１の処理によって得られたソフトウェアパイプライン化された命令スケジュール 106のカーネル部を、ループ展開部113 で複数回展開して、命令スケジュール107を得る。このループ展開数は、１度のプリフェッチ命令の実行で主記憶からキャッシュへ移動可能なキャッシュブロックの大きさをＢ、メモリ参照命令によって参照される要素の大きさをＤ、配列参照の要素の増分値をＮとすると、例えば、Ｂ/ＤとＮの最小公倍数とすればよい。

【００４１】ループの展開を行なうと、続いて、プリフェッチ命令削除部114により、ループを展開して得られた命令スケジュール 107から冗長なプリフェッチ命令を削除する。これにより、冗長なプリフェッチ命令を含まない最終的な命令スケジュール103が得られる。この冗長なプリフェッチ命令の削除に関しては、プリフェッチ命令はＢ/Ｄ回に１度発行すれば十分であることから、展開された各々のプリフェッチ命令に対して、Ｂ/Ｄ個おきにプリフェッチ命令が発行されるよう、それ以外の命令を削除する。

【００４２】以上に述べた方式では、ループ展開数が多くなる場合もあり得るので、ループ展開数を低く押えたい場合には、例えば、適当な回数だけループを展開して、先に述べたように、Ｂ/Ｄ個回の繰り返しおきにプリフェッチ命令が発行されるように、それ以外の命令を削除すれば、多少冗長なプリフェッチ命令が発行されることとなるが、展開数が大きくなるのを防ぐことができる。

【００４３】上記説明におけるプリフェッチ命令削除部114の動作を，図８に示すフローチャートを参照しつつ

説明する。まず、ステップ801ではキャッシュブロックの大きさを定数Ｂ、参照対象要素の大きさを定数Ｄとして設定する。ステップ802では、未処理プリフェッチ命令が残っているかどうかを判定し、あればステップ803へ制御を移し、なければ処理を終了する。ステップ803では、図１におけるループ展開部113によってコピーされた同一の未処理プリフェッチ命令を順に変数PFi (0≦i≦n)に記録する。ステップ804では、0≦i≦nに対して、(１をＢ/Ｄで割った余り：(ｉ mod (Ｂ/Ｄ)) ≠ ０)ならば、すなわちｉがＢ/Ｄの整数倍でなければプリフェッチ命令PFiを削除し、ステップ802へ制御を移して次のプリフェッチ命令を処理する。これにより、プリフェッチ命令がＢ/Ｄ回の繰り返しおきに発行されることになる。

【００４４】次に、方式３によりループをスケジュールする場合の実施例を説明する。方式３によりループをスケジュールする場合には、まず、方式１と同様にプリフェッチ命令生成部109で、入力となる中間語102から、キャッシュミスを起こす可能性の高いメモリ参照命令に対するプリフェッチ命令を生成し、プリフェッチ命令を加えた中間語104を得る。

【００４５】次に、プリフェッチ命令置換部110において、プリフェッチ命令生成部109で生成した複数のプリフェッチ命令を組にして仮想的なプリフェッチ命令に置き換える。この置き換えは、１度のプリフェッチ命令の実行で主記憶からキャッシュへ移動可能なキャッシュブロックの大きさをＢ、メモリ参照命令によって参照される要素の大きさをＤ、中間語104に含まれるプリフェッチ命令の数をＭとすると、例えば、M/(Ｂ/Ｄ)以上の最小の整数個の仮想的なプリフェッチ命令を作成し、Ｂ/Ｄ個毎のプリフェッチ命令と1つの仮想的なプリフェッチ命令を対応させる。仮想的なプリフェッチ命令を作成すると、中間語104中の元のプリフェッチ命令を削除し、新たに生成した仮想的なプリフェッチ命令を加える。

【００４６】次に、依存グラフ作成部111では、中間語104を入力として依存グラフ105を生成する。この場合には、方式１および２の場合と異なり、仮想的なプリフェッチ命令とメモリ参照命令の間には依存は設けない。続いて、依存グラフ105を入力として、ソフトウェアパイプライニング部112でループに対してソフトウェアパイプライニングを適用し、ソフトウェアパイプライン化された命令スケジュール106を得る。方式３では方式１および２の場合と異なり、プリフェッチ命令と対応するメモリ参照命令との間に依存関係を設けないので、ソフトウェアパイプライニングを適用する際の命令配置の自由度が高くなる。

【００４７】次に、ソフトウェアパイプライン化された命令スケジュール106 をループ展開部113で複数回展開し、命令スケジュール107を得る。この展開数は、方式２の場合と同様に、１度のプリフェッチ命令の実行で主記憶からキャッシュへ移動可能なキャッシュブロックの大きさをＢ、メモリ参照命令によって参照される要素の大きさをＤ、配列参照の要素の増分値をＮとすると、例えば、Ｂ/ＤとＮの最小公倍数とすればよい。ループ展開部113によるループ展開処理が終了すると、得られた命令スケジュール107に含まれる仮想的なプリフェッチ命令を、プリフェッチ命令復元部115においてプリフェッチ命令置換部110で置き換えた対応するプリフェッチ命令に復元する。ある仮想的プリフェッチ命令VPに、ｎ個のプリフェッチ命令 PF1,PF2,...,PFnが対応しており、仮想的プリフェッチ命令VPが，ループ展開部113によってｍ個の仮想的プリフェッチ命令VP1,VP2,...,VPmに展開されているものとすると、この復元処理は、例えば、以下のように行なわれる。

【００４８】n=Ｂ/Ｄの場合、ｊ=ｉ mod (Ｂ/Ｄ)とすると、VPiをPFjに置き換える。ｎ<; Ｂ/Ｄの場合、ｊ=ｉ mod (Ｂ/Ｄ)とすると、１ ≦ ｊ ≦ ｎならばVPiをPFjに置き換え、ｎ <; ｊならば、VPiを削除する。

【００４９】これによって、元のプリフェッチ命令からなる命令スケジュール108が得られる。

【００５０】次に、プリフェッチアドレス調整部116 によって、プリフェッチ命令によるデータの転送が完了するのに十分先の繰り返しでデータがプリフェッチされるよう、命令スケジュール108のプリフェッチ命令の参照対象のアドレスを調節し、冗長なプリフェッチ命令を含まない命令スケジュール103を得る。

【００５１】このアドレスの調節は、例えば、配列Ｘに対して、

FETCH X[i]

というプリフェッチ命令があった場合、次のように行なわれる。

【００５２】すなわち、スケジュールされたループの１回あたりの実行に要するサイクル数をＬ、プリフェッチ命令によって主記憶からキャッシュへ対象データのキャッシュブロックを転送するのに要するサイクル数をＭとすると、M/L+(Ｂ/Ｄ)より以上の最小の整数回先繰り返しで参照する配列要素をプリフェッチするようにすれば良い。

【００５３】すなわち、この繰り返し数をαとすると、上記のプリフェッチ命令の参照アドレスを、

FETCH X[i+α]

のように調節すれば良い。

【００５４】以下、上記方式３におけるプリフェッチ命令置換部110とプリフェッチ命令復元部115の処理について、フローチャートを参照しつつ説明する。

【００５５】図９は、図１におけるプリフェッチ命令置換部110の動作フローチャートである。まず、ステップ901では、キャッシュブロックの大きさを定数Ｂ、参照対象要素の大きさを定数Ｄ、プリフェッチ命令の数を記録

するための変数ｎの値を０として設定する。ステップ902では、プリフェッチ命令が残っているかどうか判定し、残っていればステップ903へ制御を移し、残っていなければ処理を終了する。ステップ903では、変数ｎの値が０かどうか判定し、真であればステップ904へ制御を移し、偽であればステップ906へ制御を移す。ステップ904では、新たに仮想的なプリフェッチ命令を生成し、これを変数VPFに記憶する。ステップ905では変数VPFに記憶された仮想的プリフェッチ命令を中間語列に挿入する。

【００５６】ステップ906ではプリフェッチ命令を選択し、変数PFに記憶する。ステップ907では変数PFに記録したプリフェッチ命令と変数VPFに記録した仮想的なプリフェッチ命令を対応させる。ステップ908では中間語列から変数PFに記録したプリフェッチ命令を削除する。ステップ909では変数ｎの値を１だけ増す。ステップ910では、ｎの値がＢ/Ｄに等しいかどうか判定し、真であればステップ911へ制御を移し、偽であればステップ902へ制御を移して次のプリフェッチ命令を処理する。ステップ911では、ｎの値を０に設定し、ステップ902へ制御を移して次のプリフェッチ命令を処理する。これによって、プリフェッチ命令がＢ/Ｄ個に１個の割合で仮想的プリフェッチ命令に置き換えられる。

【００５７】図１０は、図１におけるプリフェッチ命令復元部115の動作フローチャートである。まず、ステップ1001ではキャッシュブロックの大きさを定数Ｂ、参照対象要素の大きさを定数Ｄとして設定する。ステップ1002では仮想的プリフェッチ命令が残っているかどうか判定し、残っていればステップ1003へ制御を移し、残っていなければ処理を終了する。ステップ1003では、図１におけるループ展開部113によってコピーされた同一の仮想的プリフェッチ命令を順に変数VPi(0≦i＜m)に記憶する。ステップ1004では、VPiに対応する元のプリフェッチ命令を変数PFj(0 ≦j＜n)に記憶する。

【００５８】ステップ1005では、プリフェッチ命令PFjの数ｎがＢ/Ｄであるかどうかを判定し、真であればステップ1006へ、偽であればステップ1007へ制御を移す。ステップ1006では、各VPiに対して、j＝i mod (Ｂ/Ｄ)とすると、VPiをPFjに置き換え、制御をステップ1002へ移して次の仮想的プリフェッチ命令を処理する。ステップ1007では、各VPiに対して、j＝i mod (Ｂ/Ｄ)とすると、0≦j＜nならばVPiをPFjに置き換え、n≦jならばVPiを削除して、制御をステップ1002へ移して次の仮想的プリフェッチ命令を処理する。これによって、仮想的なプリフェッチ命令が元のプリフェッチ命令に復元されると共に、各プリフェッチ命令がＢ/Ｄ回の繰り返しが１度実行されるようになる。

【００５９】図１１は、図１におけるプリフェッチアドレス調整部116の動作フローチャートである。まず、ステップ1101ではキャッシュブロックの大きさを定数Ｂ、参照対象要素の大きさを定数Ｄ、ループ１回あたりの実行サイクルをＬ、主記憶からキャッシュへデータを転送するのに要するサイクル数をＭ、プリフェッチ命令を先行して発行すべき繰り返し数αをＭ/Ｌ+(Ｂ/Ｄ)以上の最小の整数として設定する。ステップ1102では、未処理プリフェッチ命令が残っているかを判定し、残っていればステップ1103へ制御を移し、残っていなければ処理を終了する。ステップ1103では、未処理のプリフェッチ命令を選択し、変数PFへ記録する。ステップ1104では、変数PFへ記憶したプリフェッチ命令で参照するアドレスをα回先の繰り返しで参照するアドレスへ変更する。これによって、メモリ参照命令が発行されるよりも十分先にプリフェッチ命令が発行されることになり、メモリ参照による待ちを隠蔽することができる。

【００６０】続いて、具体例を用いて各方式の一実施例によるスケジューリングの効果について説明する。図１２は実施例の説明に用いるFORTRANプログラムのループの例である。このプログラムのループ本体から、例えば図１３に示すような中間語が構成される。以下では、この中間語を入力とした場合の各方式によるプリフェッチ命令のスケジュールの例を示す。

【００６１】図１３の例では、命令1301、1302、1303でメモリ参照が行なわれるが、このうち命令1301と命令1303が参照するアドレスは同一であるので、配列Ｘおよび配列Ｙそれぞれについて１つづつプリフェッチ命令を作成する。なお、この例ではメモリ参照命令およびプリフェッチ命令と演算命令を並列に実行できるスーパスカラ型のプロセッサを仮定する。ただし、本発明はスーパスカラ型のプロセッサのみに適用可能というわけではなく、逐次型のプロセッサや超多長命令形式（VLIW）のプロセッサについても適用可能である。なお、以下の例では、一度のプリフェッチで４回分の繰り返しで使用するデータをキャッシュに転送可能であり、主記憶からキャッシュへのデータの転送には50サイクル必要であると仮定する。

【００６２】方式１：

(１)プリフェッチ命令の生成

配列Ｘおよび配列Ｙについて、プリフェッチ命令を生成する。プリフェッチ命令を付加した中間語は図１４に示すようになる。この図で、命令1401および命令1402はそれぞれ配列Ｘおよび配列Ｙに対するプリフェッチ命令である。

【００６３】(２)依存グラフの生成

図１５にプリフェッチ命令を加えた中間語に対する依存グラフを示す。この図で、ノードは命令を表し、ノード間のエッジは依存関係を表している。各エッジの右に付加している数字は命令間を離すべきサイクル数を表している。この図に示すように、配列Ｘに対するプリフェッチ命令1501と配列Ｘに対するロード命令1503との間、および、配列Ｙに対するプリフェッチ命令1502と配列Ｙの

ロード命令1504との間に主記憶からキャッシュへデータを転送するのに必要な50サイクルの遅延を持つ依存関係を設ける。

【００６４】(3)ソフトウェアパイプライニング

図１５の依存グラフに対して、ソフトウェアパイプライニングを適用する。ソフトウェアパイプライン化されたスケジュールは図１６のようになる。図１６に示すスケジュールは、ループの初期化を行なうプロローグ部1601、ループの繰り返しを行なうカーネル部1602、ループの終了処理を行なうエピローグ部1603からなり、図１６の各エントリは各サイクルに対応する命令スロットを表している。プリフェッチ命令は命令スロット1604および1605に割り当てられており、ソフトウェアパイプライニングによって、対応するメモリ参照命令の10回前の繰り返しで実行されるようにスケジュールされている。以上のように、ソフトウェアパイプライニングによって各命令間の依存を満たすようなスケジュールが得られたので、メモリ参照に伴う待ちが除去されることになる。

【００６５】方式２：上記方式１の実施例では、１回の繰り返しにつき２つのプリフェッチ命令が発行されることになる。プリフェッチ命令では４回分の繰り返しで利用するデータをプリフェッチすることが可能なので、毎回プリフェッチ命令を発行するのは無駄である。そこで、方式２では以下の処理を方式１の結果に適用することで、無駄なプリフェッチ命令の発行を抑制する。

【００６６】(4)ループ展開

方式１の(3)で構成されたソフトウェアパイプライン化されたループのカーネル部を展開する。本実施例では、一度のプリフェッチで４回分の繰り返しで参照するデータをキャッシュへ転送可能であると仮定しているので、繰り返し４回に１回の割合でプリフェッチ命令を発行すれば良い。そこで、カーネルを４回展開すると、図１７に示すようなスケジュールが得られる。図１７に示すスケジュールは、プロローグ部1701、展開されたカーネル部1702、及びエピローグ部1703からなる。カーネル部1702において配列Ｘに対するプリフェッチ命令は、命令スロット1704、1706、1708、1710に展開され、配列Ｙに対するプリフェッチ命令は、命令スロット1705、1707、1709、1711に展開されている。

【００６７】(5)冗長なプリフェッチの削除

ループ展開された図１７の命令スケジュールに対して、プリフェッチ命令の発行が４回の繰り返し毎に行なわれるよう配列Ｘおよび配列Ｙに対する冗長なプリフェッチ命令を削除する。これによって無駄なプリフェッチ命令の発行が抑制され、図１８に示すようなスケジュールが得られる。図１８において、プロローグ部1802では冗長なプリフェッチ命令1805、1806、1807、1808、1810、1811が削除され、配列Ｘに対しては命令スロット1804、配列Ｙに対しては命令スロット1809のそれぞれのプリフェッチ命令によって、４回分の繰り返しのデータが無駄なくプリフェッチされるようになる。

【００６８】方式３：方式３では、無駄なプリフェッチ命令を発行しないことを考慮して以下のようにプリフェッチ命令のスケジュールを行なう。

【００６９】(1)プリフェッチ命令の生成

プリフェッチ命令の生成は方式１の場合と同様に行なう。

【００７０】(2)プリフェッチ命令の置換と依存グラフの作成

上記(1)で生成した複数のプリフェッチ命令を命令をまとめて仮想的なプリフェッチ命令を生成し、依存グラフを構成する。結果の依存グラフを図１９に示す。図１９に示すように、配列Ｘに対するプリフェッチ命令1901と、配列Ｙに対するプリフェッチ命令1902をまとめて仮想的なプリフェッチ命令1903に置き換える。方式１や方式２の場合と異なり、方式３では仮想的なプリフェッチ命令と対応するメモリ参照命令の間に依存関係は設けない。

【００７１】(3)ソフトウェアパイプライニング

仮想的なプリフェッチ命令を加えたループ本体にソフトウェアパイプライニングを適用する。この結果、例えば、図２０に示すようなソフトウェアパイプライン化されたスケジュールが得られる。図２０に示すスケジュールは、プロローグ部2001、カーネル部2002、及びエピローグ部2003からなる。カーネル部2002の命令発行スロット2004には仮想的なプリフェッチ命令がスケジュールされている。

【００７２】(4)ループ展開

方式２の場合と同様に上記(3)で構成したソフトウェアパイプライン化されたスケジュールのカーネル部を４回分展開する。これにより、図２１のようなスケジュールが得られる。図２１にしめすスケジュールは、プロローグ部2101、展開されたカーネル部2102、及びエピローグ部2103からなる。ループ展開によってカーネル部2103にスケジュールされた仮想的なプリフェッチ命令は命令発行スロット2104、2105、2106、2107に展開されている。

【００７３】(5)プリフェッチ命令の復元

図２１のカーネル部2102の命令スロット2104、2105、2106、2107に展開された仮想的プリフェッチ命令を元のプリフェッチ命令に置き換える。この結果を図２２に示す。図２１の命令スロット2104、2105、2106、2107に展開された仮想的プリフェッチ命令は配列Ｘおよび配列Ｙに対するプリフェッチ命令を置き換えたものであるので、各々の配列に対するプリフェッチ命令が４回の繰り返しに１回発行されるように、図２２の命令スロット2204および命令スロット2206にプリフェッチ命令を挿入しする。この場合、元のプリフェッチ命令数が一度のプリフェッチで主記憶からキャッシュへ転送可能なデータを参照する繰り返し数に満たないので、展開された仮想的なプリフェッチ命令に対応する命令スロット2205と2207

については空きスロットとする。

【００７４】(6)プリフェッチアドレスの調節

プリフェッチ命令の発行とプリフェッチ命令によってキャッシュに転送されるデータを参照する命令の発行とが、主記憶からキャッシュへのキャッシュブロックの転送が終了するのに十分なサイクル数だけ離れて行なわれるようプリフェッチ対象のアドレスを調整する。ここで、主記憶からキャッシュへキャッシュブロックを転送するのに要するサイクルは50サイクルで、１回あたりの繰り返しに要するサイクルは４サイクルであるので、図２２に示すように、ここでは１４回先の繰り返しで参照されるデータをプリフェッチするようにプリフェッチ命令の参照先を変更している。

【００７５】以上の説明により、図１の中間語102を入力としてスケジューラ101によりプリフェッチ命令を含んだ命令スケジュール103を作成することができる。すなわち、ループ繰り返しにおいて、データの参照が連続的でない場合には方式１を用いることで、プリフェッチ命令と対応するメモリ参照命令を、主記憶からキャッシュへデータを転送するのに必要なサイクル数だけ離して発行することができるので、メモリ参照に伴う待ちを隠蔽することができる。また、データの参照が連続的な場合には、方式２および３を用いることで、冗長なプリフェッチ命令の発行を抑制することができる。さらに、方式３では方式２と比較して、仮想的なプリフェッチ命令とメモリ参照命令との間に依存関係を設けないので、命令配置の自由度が高くなり、また、カーネル部の展開を考慮してソフトウェアパイプライニングを適用するので命令間の依存などによる待ちの発生を低く押えることができる。

【００７６】

【発明の効果】本発明によれば、プリフェッチ命令を効果的にスケジュールし、プログラムの実行時のメモリ参照等による待ちを減少することができる。これにより計算機プログラムの実行の高速化に効果がある。

【００７７】すなわち、本発明の方法によれば、メモリの参照が連続的でない場合には、方式１によってプリフェッチ命令と対応するメモリ参照命令の間を十分離してソフトウェアパイプライニングを適用することができる。また、メモリの参照が連続的である場合には、方式２によってソフトウェアパイプライニングを適用した後で命令の削除を行なうか、方式３によって複数のプリフェッチ命令を仮想的なプリフェッチ命令に置き換えてソフトウェアパイプライニングを適用し、その後、元のプリフェッチ命令に復元することで無駄なプリフェッチ命令を発行を抑制し効率的にスケジュールすることができる。

【図面の簡単な説明】

【図１】プリフェッチ命令をスケジュールする命令スケジューラの構成図である。

【図２】本発明を実施する計算機システムの例である。

【図３】本発明の対象とする計算機システムの例である。

【図４】方式1による命令スケジューラの構成図である。

【図５】方式2による命令スケジューラの構成図である。

【図６】方式3による命令スケジューラの構成図である。

【図７】プリフェッチ命令生成部のフローチャートである。

【図８】プリフェッチ命令削除部のフローチャートである。

【図９】プリフェッチ命令置換部のフローチャートである。

【図１０】プリフェッチ命令復元部のフローチャートである。

【図１１】プリフェッチアドレス調節部のフローチャートである。

【図１２】FORTRANソースプログラムの例である。

【図１３】中間語の例である。

【図１４】プリフェッチ命令を含んだ中間語の例である。

【図１５】方式1の依存グラフの例である。

【図１６】方式1のソフトウェアパイプライン化されたスケジュールの例である。

【図１７】方式2の展開したスケジュールの例である。

【図１８】方式2の冗長なプリフェッチ命令削除を行なったスケジュールの例である。

【図１９】方式3の依存グラフの例である。

【図２０】方式3のソフトウェアパイプライン化されたスケジュールの例である。

【図２１】方式3の展開したスケジュールの例である。

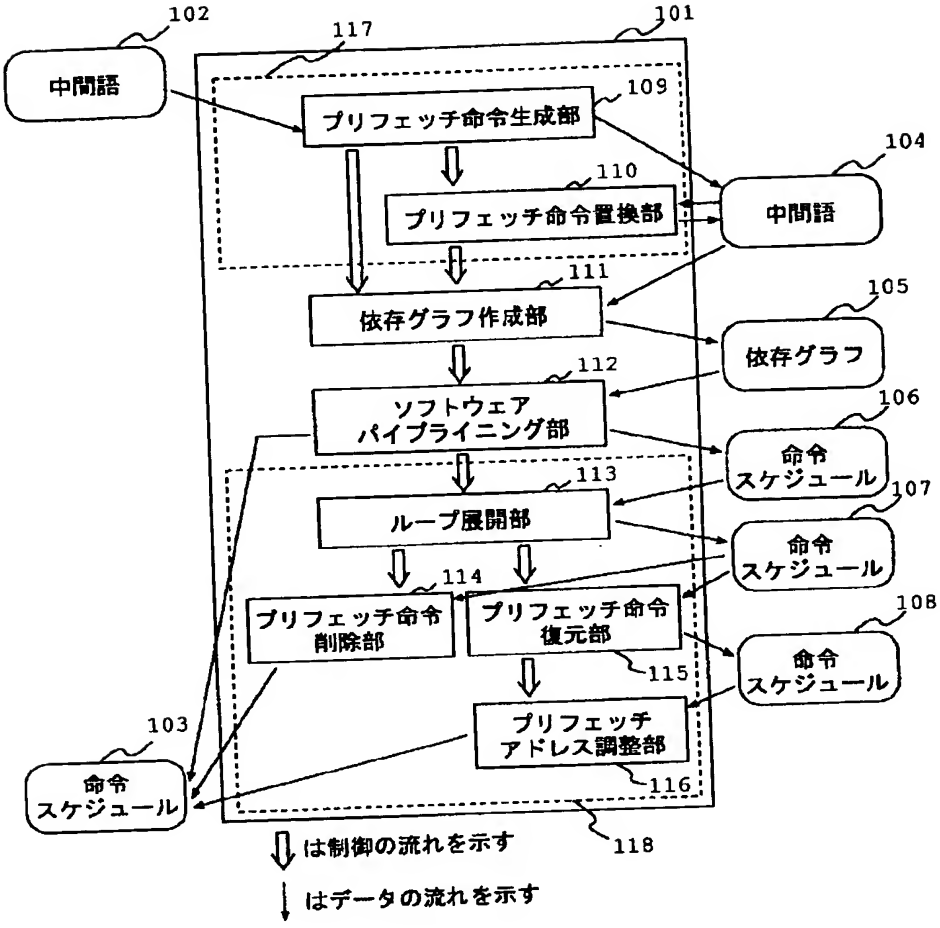
【図２２】方式3のプリフェッチ命令の置換えを行なったスケジュールの例である。

【符号の説明】

101：ループに対するスケジューリング処理部
109：プリフェッチ命令生成部
110：プリフェッチ命令置換部
111：依存グラフ作成部
112：ソフトウェアパイプライニング部
113：ループ展開部
114：プリフェッチ命令削除部
115：プリフェッチ命令復元部
116：プリフェッチアドレス調整部

【図１】

図１



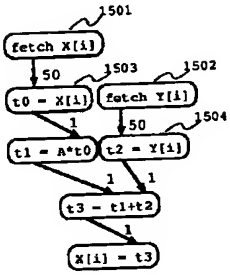
【図１２】

図12

```
DO 10 I=0, N
   X(I) = A*X(I)+Y(I)
10 CONTINUE
```

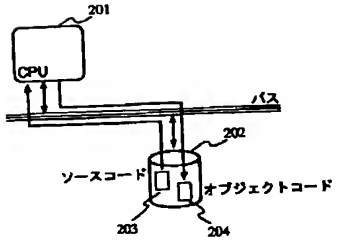
【図１５】

図15



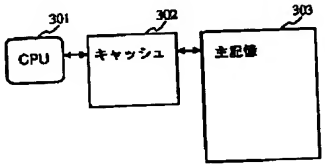
【図２】

図2



【図３】

図3



【図１３】

図13

```
t0=X[i]～1301
t1=A*X[i]
t2=Y[i]～1302
t3=t1+t2
X[i]=t3～1303
```

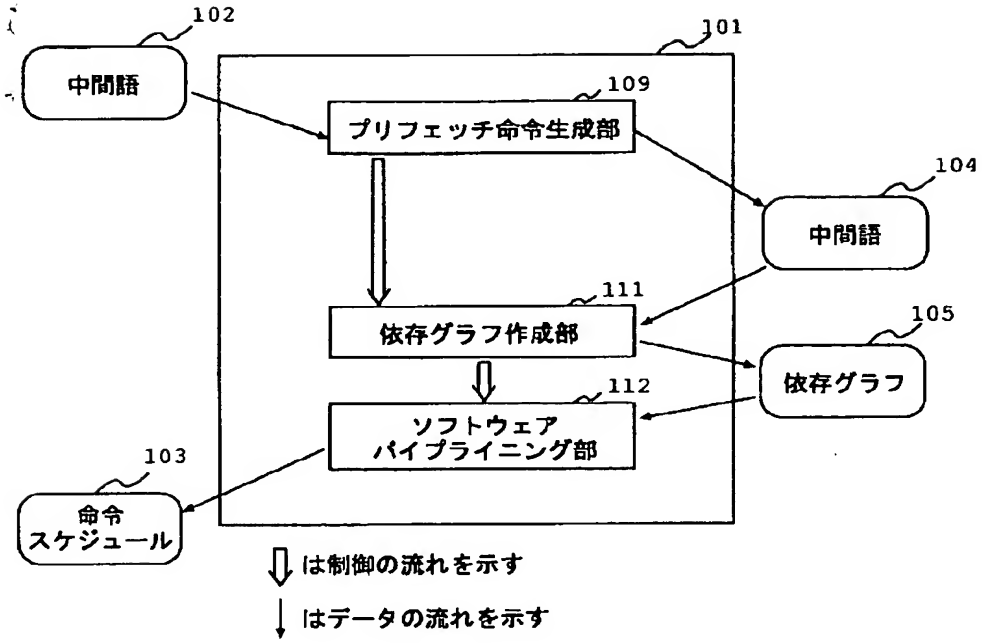
【図１４】

図14

```
fetch X[i]～1401
fetch Y[i]～1402
t0=X[i]
t1=A*X[i]
t2=Y[i]
t3=t1+t2
X[i]=t3
```

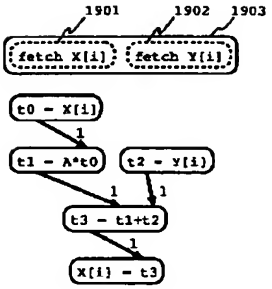
【図４】

図４



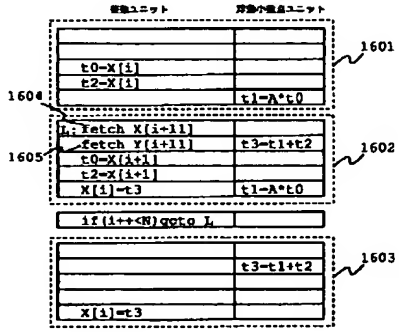
【図１９】

図19



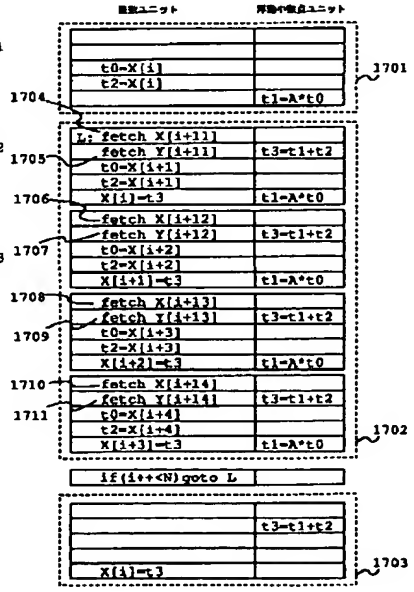
【図１６】

図16



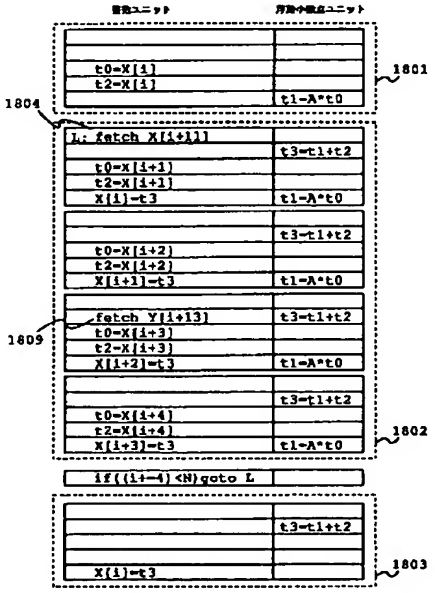
【図１７】

図17



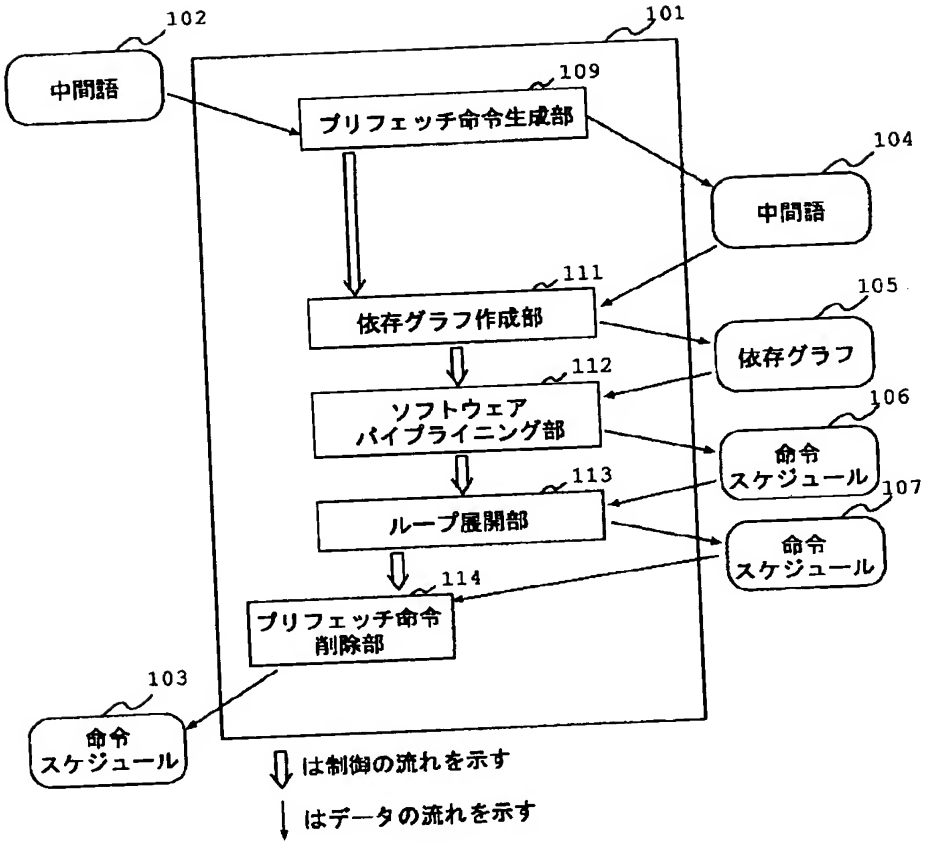
【図１８】

図18



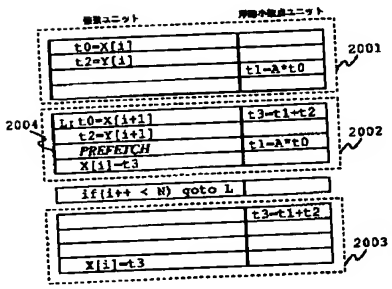
【図５】

図５


102
101
中間語
109
プリフェッチ命令生成部
104
中間語
111
依存グラフ作成部
105
依存グラフ
112
ソフトウェア
パイプライニング部
106
命令
スケジュール
113
ループ展開部
107
命令
スケジュール
114
プリフェッチ命令
削除部
103
命令
スケジュール
は制御の流れを示す
はデータの流れを示す

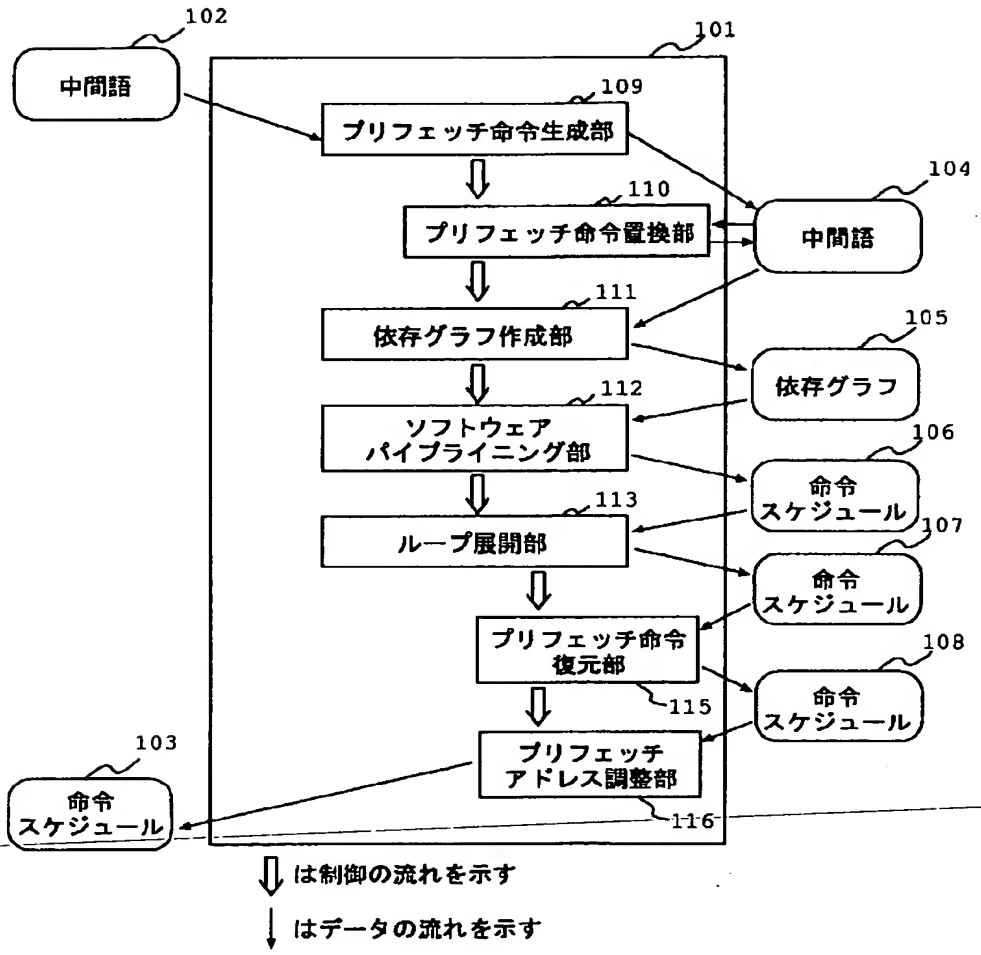
【図２０】

図 20


t0=X[i]
t2=Y[i]
t1=A*t0
2001
L: t0=X[i+1]
t2=Y[i+1]
PREFETCH
X[i]=t3
t3=t1+t2
t1=A*t0
2002
2004
if(i++ < N) goto L
t3=t1+t2
X[i]=t3
2003

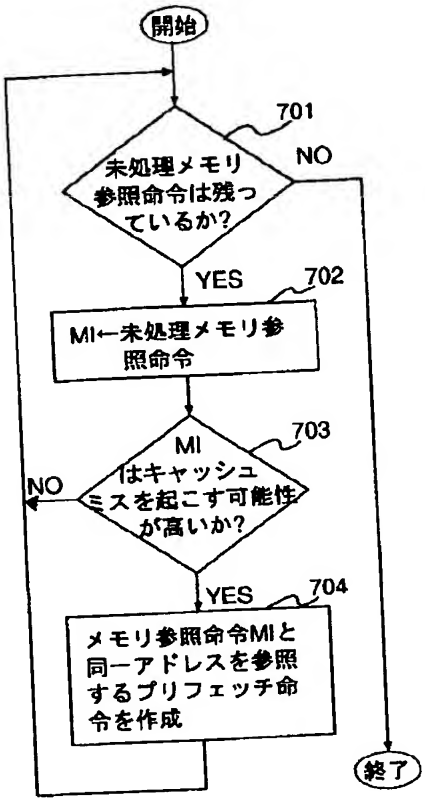
【図６】

図６


102
中間語
101
109
プリフェッチ命令生成部
110
プリフェッチ命令置換部
104
中間語
111
依存グラフ作成部
105
依存グラフ
112
ソフトウェア
パイプライニング部
106
命令
スケジュール
113
ループ展開部
107
命令
スケジュール
プリフェッチ命令
復元部
108
命令
スケジュール
115
103
命令
スケジュール
プリフェッチ
アドレス調整部
116
は制御の流れを示す
はデータの流れを示す

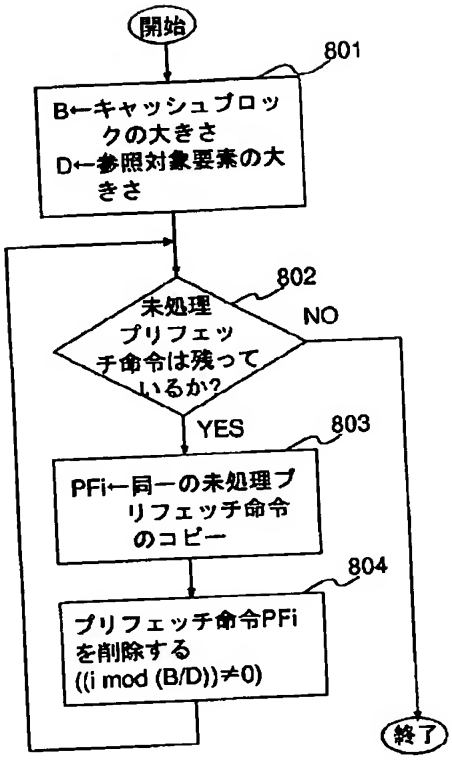
【図7】

図7


開始
701
未処理メモリ参照命令は残っているか?
NO
YES
702
MI←未処理メモリ参照命令
703
MI はキャッシュミスを起こす可能性が高いか?
NO
YES
704
メモリ参照命令MIと同一アドレスを参照するプリフェッチ命令を作成
終了

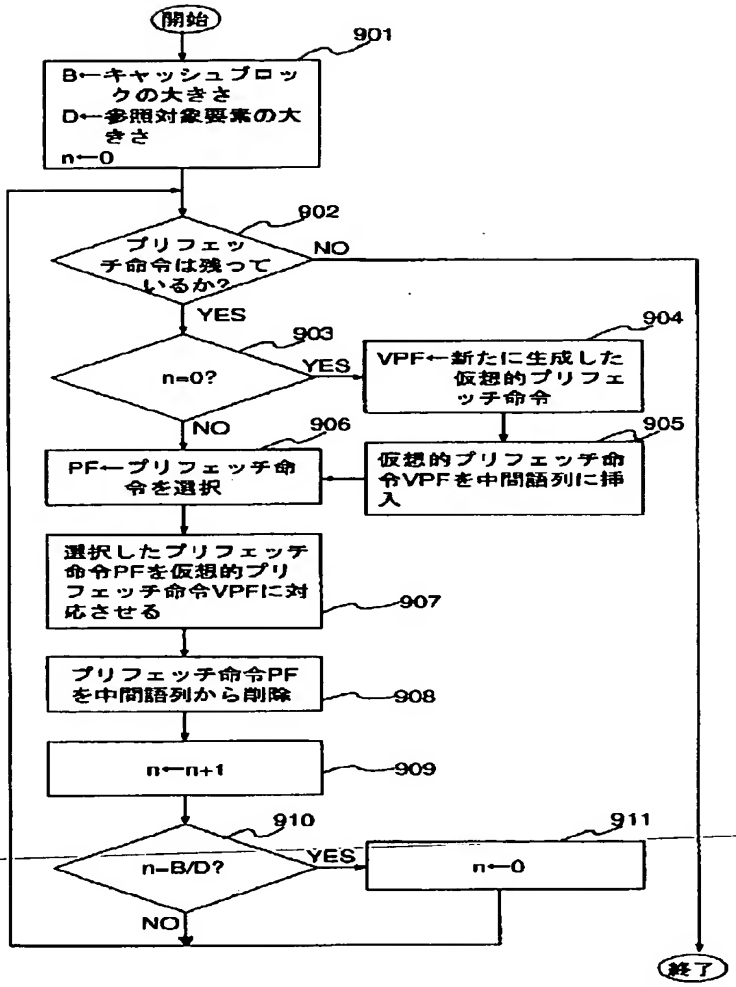
【図8】

図8


開始
801
B←キャッシュブロックの大きさ
D←参照対象要素の大きさ
802
未処理プリフェッチ命令は残っているか?
NO
YES
803
PFi←同一の未処理プリフェッチ命令のコピー
804
プリフェッチ命令PFiを削除する
((i mod (B/D))≠0)
終了

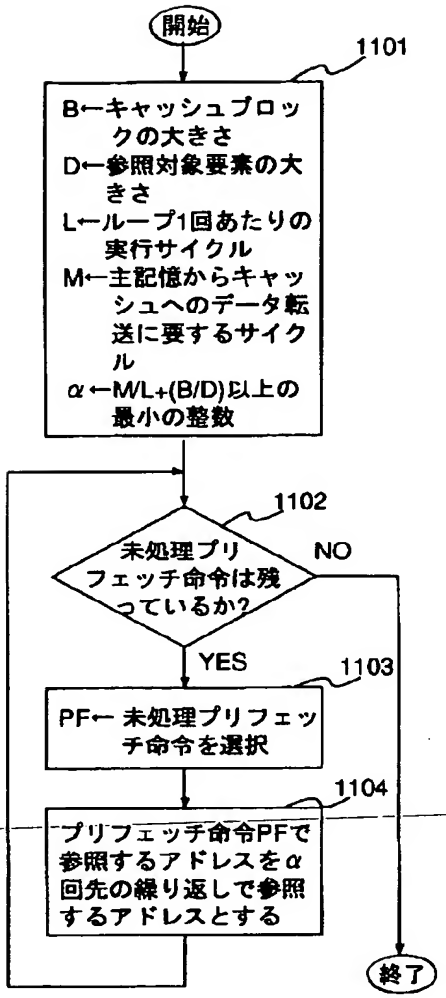
【図9】

図9


開始
901
B←キャッシュブロックの大きさ
D←参照対象要素の大きさ
n←0
902
プリフェッチ命令は残っているか?
NO
YES
903
n=0?
YES
904
VPF←新たに生成した仮想的プリフェッチ命令
NO
906
PF←プリフェッチ命令を選択
905
仮想的プリフェッチ命令VPFを中間語列に挿入
選択したプリフェッチ命令PFを仮想的プリフェッチ命令VPFに対応させる
907
プリフェッチ命令PFを中間語列から削除
908
n←n+1
909
910
n=B/D?
YES
911
n←0
NO
終了

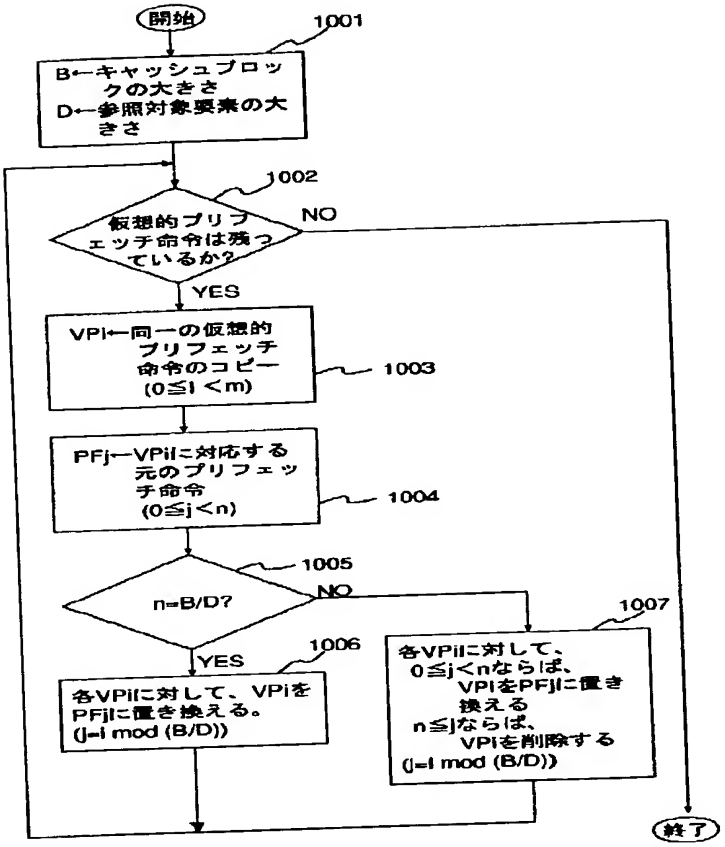
【図11】

図11


開始
1101
B←キャッシュブロックの大きさ
D←参照対象要素の大きさ
L←ループ1回あたりの実行サイクル
M←主記憶からキャッシュへのデータ転送に要するサイクル
α←M/L+(B/D)以上の最小の整数
1102
未処理プリフェッチ命令は残っているか?
NO
YES
1103
PF← 未処理プリフェッチ命令を選択
1104
プリフェッチ命令PFで参照するアドレスをα回先の繰り返しで参照するアドレスとする
終了

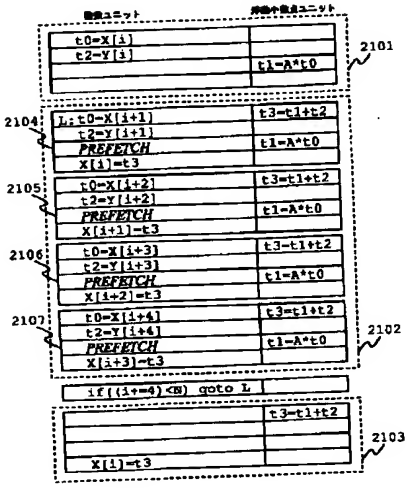
【図10】

図10



【図21】

図21



【図22】

図22